WeatherNet Weather Forecast Using ML Techniques

Dor Shabat and Yuval Rozner

BRAUDE College of Engineering, Karmiel

Capstone Project Phase B - 61998

Dr. Dan Lemberg and Mrs. Elena Kramer

February 04, 2025

Contents

Abstract **3**
Deliverables **3**
Introduction **4**
The Architecture **6**
Introduction and Architecture Overview 6
Hybrid Model Approach 6
Model Components 8
StationCNN 8
Coordinate Positional Encoding (Spatial Context) 9
Temporal Positional Encoding (Time Context) 9
Transformer Encoder 9
Fully Connected Layer (Output) 10
Window Generation and Data Pipeline 10
Training Procedure 10
HyperParameters 11
Project Workflow **12**
Research Phase 12
Development & Implementation Phase 12
Challenges and Solutions **14**
Data Preprocessing 14
Resource Constraints for AI Model Training 14
Hyperparameters 15
Deployment and User Accessibility 15
From Vision to Reality: What We Achieved **16**
Phase A Expectations vs. Achievements 16
What We Achieved 16
Predicting Weather Using an ML Model 16
Website Development 18
Closing Statement **19**

Abstract

WeatherNet introduces an innovative system for forecasting weather in Israel utilizing advanced machine learning techniques. Focusing on mid-term weather predictions, specifically temperature forecasts for Israel's northern region, the system harnesses both historical and real-time data from the Israel Meteorological Service (IMS) to develop a robust machine learning-based solution.

At its core, WeatherNet employs a hybrid architecture that combines 1D Convolutional Neural Networks (CNNs) with Transformers incorporating attention mechanisms, effectively capturing temporal trends and the relationships between geographically diverse locations. This approach adeptly identifies short-term temporal patterns, long-term dependencies, and spatial relationships among weather stations.

WeatherNet delivers reliable temperature forecasts up to three days in advance through a user-friendly web-based platform. The platform features interactive visualizations and provides insights into prediction accuracy and the underlying architecture, enabling users to explore and comprehend the machine learning methodologies driving our forecasts. By introducing a machine learning-driven approach, WeatherNet broadening the range of techniques available for accurate weather forecasting in Israel.

Keywords: *Time Series Forecasting, Israeli Weather Forecasting, Machine Learning (ML), Hybrid Architecture, Web-Based Platform.*

Deliverables

This section provides all the materials submitted as part of the WeatherNet project.

Website	weathernet-il
Github	WeatherNet Project Github
User Manual	PDF
Developer Manual	PDF
Phase A paper	PDF

Introduction

WeatherNet is an advanced weather forecasting system that utilizes machine learning to deliver precise mid-term temperature predictions for Israel. This paper, the second in a two-phase documentation process, presents the final design of the system architecture, its implementation, evaluation, and deployment. The first phase centered on research and proof of concept (POC), while this phase focuses on the fully developed product, which aligns closely with most of the original design specifications and system requirements.

The WeatherNet project was structured into two primary phases. The first phase involved extensive research on weather forecasting techniques, machine learning methodologies, and data acquisition from the Israel Meteorological Service (IMS). During this phase, we developed a proof-of-concept model to validate the feasibility of our approach. The second phase, described in this paper, transitioned from research to full-scale implementation, including the training of our machine learning model, the development of a user-friendly web interface, and the deployment of a fully functional system.

WeatherNet consists of two core components:

- **Backend Machine Learning Model**: Responsible for processing weather data, training on historical and real-time information, and generating accurate predictions.
- **Frontend Web-Based Platform**: Provides an intuitive interface for users to access forecasts, compare results with traditional meteorological predictions, and explore model performance insights.

This paper covers the detailed implementation of the system, including architecture, training procedures, performance analysis, and deployment considerations. Additionally, it is supplemented by two key documents: the **WeatherNet User Manual**, which provides end-users with guidance on using the platform, and the **WeatherNet Developer Manual**, offering in-depth technical insights for developers maintaining or extending the system.

One of the primary objectives of this phase was to ensure that the final implementation closely aligns with the requirements, testing criteria, and UML diagrams established in the **Phase A Paper**. Through rigorous validation and iterative development, we successfully delivered a system that meets these predefined standards, demonstrating the effectiveness of our approach. This document presents the culmination of our work,

highlighting what we achieved, the results, challenges encountered, and solutions that led to the deployment of a reliable and scalable weather forecasting platform.

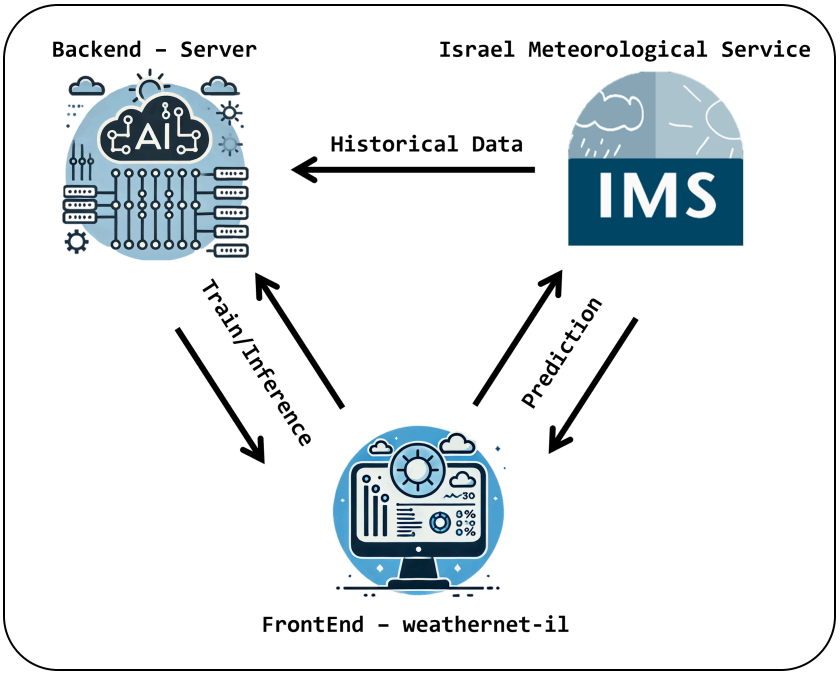

figure 01. Diagram of the WeatherNet System Architecture - Macro View

The Architecture

Introduction and Architecture Overview

The proposed **WeatherNetModel** (implemented as TargetedWeatherPredictionModel in code) is designed to predict weather parameters for a *specific station* while leveraging information from *multiple nearby stations*. The architecture addresses both the **temporal** and **spatial** dimensions of weather data:

1. **Temporal dependencies** are captured through station-specific 1D Convolutional Neural Networks (CNNs) and temporal positional encodings.
2. **Spatial dependencies** are incorporated through station-level coordinate encodings and a Transformer Encoder mechanism that attends across all stations.

This dual attention to *where* (station locations) and *when* (time windows) equips the model to accurately forecast weather metrics over different future time horizons.

Hybrid Model Approach

To adapt to varying weather dynamics at different forecasting ranges, we employ a **multi-model strategy**. Specifically, four instances of the model are trained independently, each optimized for a distinct forecast horizon:

- **0–12 hours:** Captures fast-evolving short-term weather changes.
- **12–24 hours:** Targets near-daily patterns.
- **24–36 hours:** Focuses on mid-term trends.
- **36–60 hours:** Tackles longer-range forecasts.

Although these models share the same underlying architecture, each is specialized to its respective time window. This division helps the system learn unique temporal dependencies pertinent to short-, medium-, or long-term forecasting, ultimately **improving predictive performance** across all ranges.

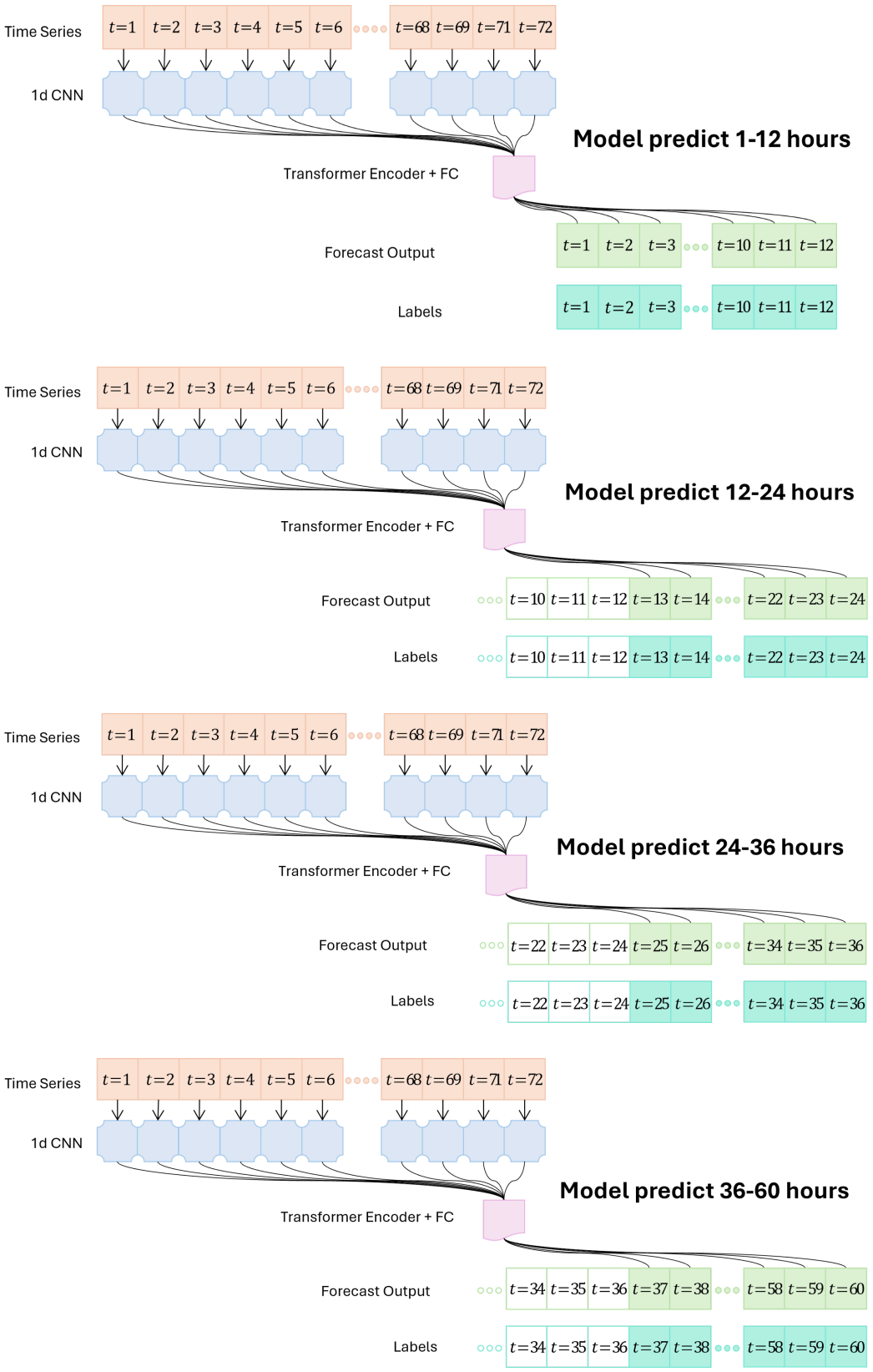

Figure 02. Diagram of the time series through the model dedicated for different prediction windows.

Model Components

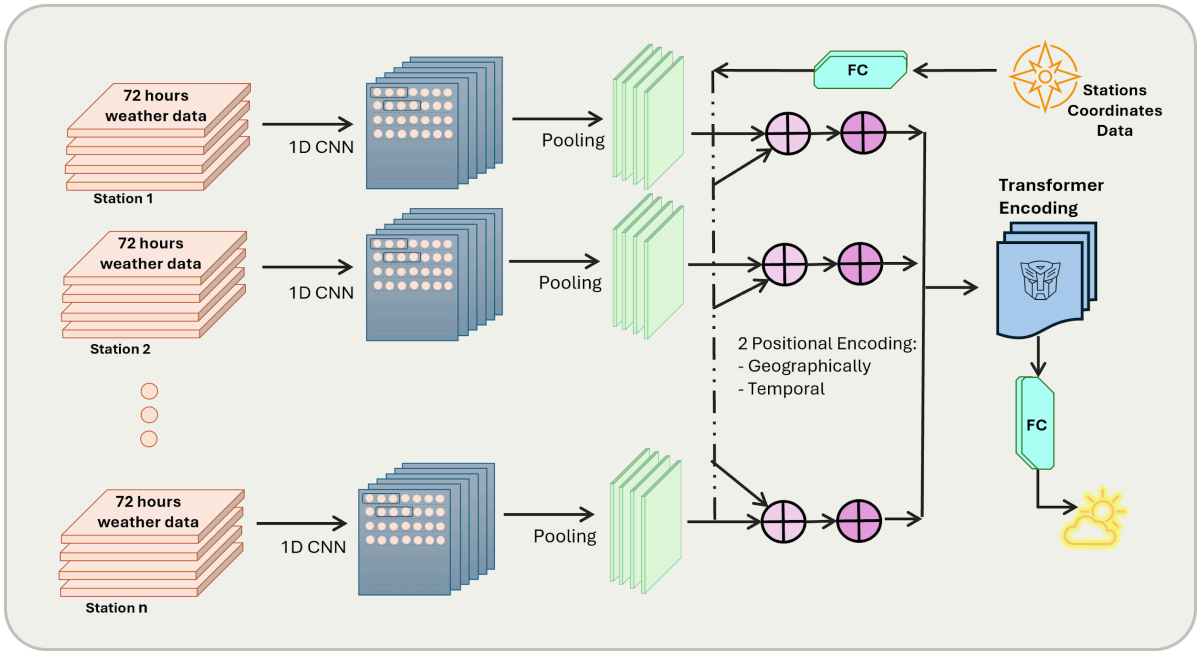

figure 03. The WeatherNet ML Architecture

StationCNN

Each weather station has its own convolutional feature extractor (StationCNN), which processes raw input time series independently:

1. **Convolution**: Each CNN applies group-wise convolutions where every feature channel is learned separately.
2. **ReLU Activation**: Nonlinear activation captures local patterns essential for weather variability.
3. **Feature Mapping**: The CNN outputs station-specific latent representations (feature_dim in the code) that encapsulate high-level temporal trends.

Formally, if xi denotes the time-series data from station i, then StationCNN transforms it into a feature map fi that highlights the distinctive temporal signature of that station.

Coordinate Positional Encoding (Spatial Context)

Geographical proximity often correlates with similar weather behavior. To incorporate *where* each station is located, we employ a coordinate-based positional encoding (CoordinatePositionalEncoding):

- **Inputs**: Station coordinates (north,east) in a specified projection (e.g., Israeli Transverse Mercator).
- **Linear Projection**: Each coordinate is mapped into half of the embedding dimension (dmodel/2), then concatenated.
- **Activation**: A ReLU nonlinearity injects modest nonlinearity into the spatial embedding.

This spatial encoding is later added to each station's temporal features to help the network attend to *which* stations are geographically influential.

Temporal Positional Encoding (Time Context)

Following the original Transformer architecture, a sinusoidal positional encoding (TemporalPositionalEncoding) is used to inject **relative and absolute time-step** information into the model's feature space. This module:

- Generates fixed sine and cosine embeddings for positions 1,2,…,max_len.
- Adds these positional embeddings to the station-specific temporal features, enabling the network to reason about *when* certain patterns occur.

Transformer Encoder

After each station's temporal features and the spatial coordinate embeddings are fused, the resulting combined embeddings are fed to a **Transformer Encoder**:

1. **Input Flattening**: The station and time dimensions are merged into a single sequence dimension.
2. **Multi-Head Attention**: The transformer layer learns to attend to interactions *across stations* and *across time steps*.
3. **Feedforward Layers**: Post-attention linear transformations capture higher-level nonlinearities.

This mechanism excels at modeling complex dependencies, ensuring that the model can learn which stations and time steps matter most for the target forecast.

Fully Connected Layer (Output)

A final linear layer maps the Transformer outputs to the **label width** (e.g., number of time steps or weather variables to predict). In practice, the model focuses on the **target station**'s final or selected time-step features, transforming them into the desired weather predictions:

prediction=Linear(TransformerOutput(stationtarget))

For example, if the label width is 1, the model predicts a single weather temperature at a specific future time. If the label width is larger, the final linear layer can output multiple future time steps or multiple weather variables simultaneously.

Window Generation and Data Pipeline

Weather data is often stored as long time series with multiple stations. To train efficiently, we generate input–label pairs using a custom sliding-window approach, implemented in the WindowGeneratorMultipleStations class:

1. **Windowing Logic**: For each station and time segment, we extract an **input window** of length input_width and pair it with a **label window** of length label_width, offset by a chosen shift.
2. **Batch Construction**: Multiple windows are stacked into batches. Each batch has shape [batch_size,num_stations,time_steps,num_features]
3. **Target Station Specification**: Although the model processes data from *all* stations, labels are taken from the **target station** to focus the final predictions on the station of interest.

This modular windowing system simplifies experiments with different sequence lengths, forecast horizons, and station combinations.

Training Procedure

Training is orchestrated in train.py and proceeds as follows:

1. **Loss Function**: The model uses Mean Squared Error (MSE) loss to measure forecast accuracy.
2. **Optimizer**: An Adam optimizer with an initial learning rate (hyperparameter in parameter.py) is employed for stable and efficient convergence.
3. **Learning Rate Scheduler**: A ReduceLROnPlateau scheduler dynamically decreases the learning rate upon observing plateaus in validation loss, enabling better fine-tuning in later epochs.
4. **Early Stopping**: Training halts if validation loss fails to improve for a set number of epochs (e.g., **10 patience**).
5. **Checkpointing**: Both "latest" and "best" model states are saved. This ensures resilience in case of interruptions and also preserves the model instance with the best validation performance.

This pipeline is adaptable to large datasets and can be resumed from saved checkpoints, promoting experimental flexibility.

HyperParameters

Several hyperparameters were tested to optimize the performance of our model, with their configurations recorded in parameter.py. These hyperparameters include the learning rate, batch size, sequence length, and model-specific parameters such as the number of Transformer layers, attention heads, and hidden dimensions. To determine the best settings, we conducted extensive experiments across different forecast horizons, tuning each model instance separately for short-term, mid-term, and long-term predictions. We observed that smaller learning rates improved stability for longer forecast horizons, while batch size variations influenced convergence speed. The combination of CNN and Transformer components required careful balancing of filter sizes and embedding dimensions to maximize both temporal and spatial feature extraction. The results per model configuration, as well as the impact of each hyperparameter, are detailed in parameter.py, allowing for reproducibility and further fine-tuning.

Project Workflow

The WeatherNet project evolved through a structured workflow across two semesters, transitioning from research and planning to full implementation and deployment.

Research Phase

In the first semester, our primary focus was gaining an in-depth understanding of weather forecasting and machine learning methodologies. We conducted extensive research through multiple channels:

- Established a connection with the **Israel Meteorological Service (IMS)** to access real-time and historical weather data.
- Researched traditional weather forecasting methods to understand the strengths and limitations of physics-based models.
- Studied the impact of various weather components (temperature, humidity, wind speed, etc.) on forecasting accuracy.
- Explored existing machine learning solutions for **time-series forecasting**, both in general applications and specifically for weather prediction.
- Reviewed academic papers and industry practices to identify optimal machine learning architectures for weather forecasting.

To validate our research findings, we developed a **proof-of-concept mini-system** that tested our initial model's feasibility. This phase concluded with the publication of our **Phase A paper** documenting our findings and initial implementation.

Development & Implementation Phase

In the second semester, we shifted from research to full-scale development and model optimization. This phase included:

- **Data Processing:** Handling vast amounts of meteorological data, applying preprocessing techniques like normalization and feature engineering.
- **Model Development:** Constructing and refining our machine learning model, incorporating **CNNs and Transformers** with attention mechanisms to enhance prediction accuracy.

- **Hyperparameter Tuning:** Iteratively adjusting parameters to improve model performance and reduce prediction errors.
- **Frontend & Backend Development:** Developing a **React-based web interface** and integrating it with our backend prediction module to deliver real-time forecasts.

Finally, we successfully **deployed a live version of WeatherNet**, allowing users to access real-time predictions through an intuitive platform. This phase concluded with the publication of our **Phase B paper**, along with the release of both the **User Manual and Developer Manual** to document system usage and maintenance.

This structured approach enabled us to transition from a conceptual framework to a fully operational, machine-learning-driven weather forecasting platform.

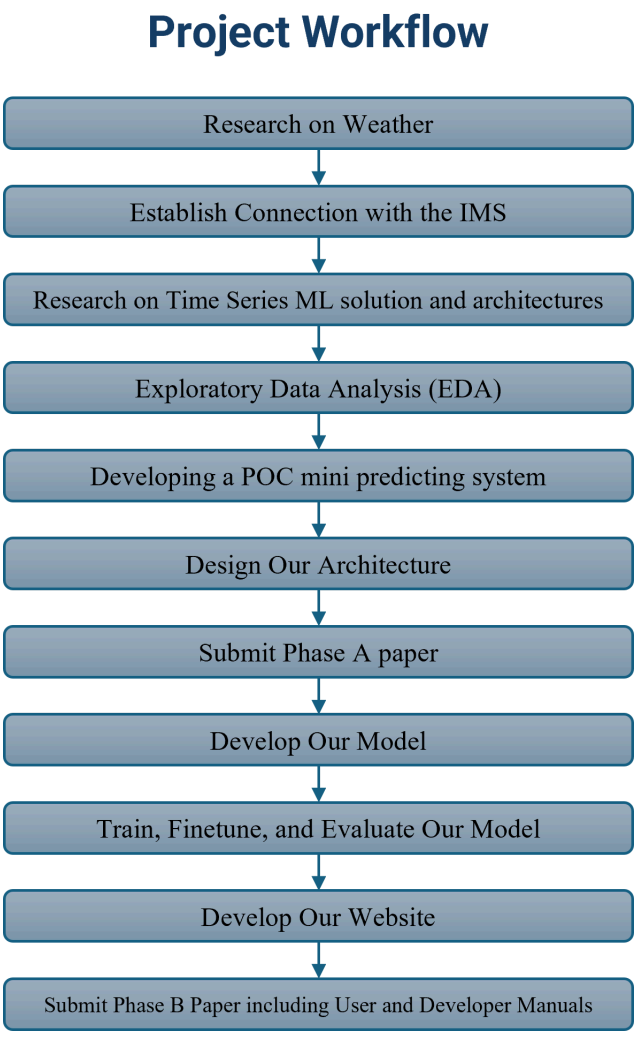

figure 04. WeatherNet Project Workflow

Challenges and Solutions

Developing WeatherNet came with multiple challenges, from handling incomplete data to optimizing model performance and ensuring seamless deployment. Each stage of the project required creative problem-solving to overcome technical and resource-related obstacles. This section outlines the key challenges we faced and the solutions that made our system more robust and efficient.

Data Preprocessing

We sourced 20 years of historical weather data from the Israeli Meteorological Service (IMS) API and faced two major issues:

1. **Incomplete Data:** Some stations had missing records for specific time periods due to gaps in measurements.
2. **Time Synchronization Issues:** Data from different stations were not synchronized, leading to discrepancies in timestamps.

Solution: We developed a comprehensive Python-based preprocessing pipeline that:

1. **Filled Missing Data:** Used interpolation techniques to estimate missing values where appropriate.
2. **Synchronized Timestamps:** Aligned all station data to a common time axis, eliminating discrepancies and ensuring consistency across stations.

These steps ensured that the dataset was clean, consistent, and ready for the preprocessing stage.

Resource Constraints for AI Model Training

Training complex machine learning models on large datasets demands substantial computational resources. Our initial hardware was insufficient, limiting our ability to train sophisticated models efficiently.

Solution: To address this limitation, we subscribed to **Google Colab Pro**, which provided access to enhanced computing resources, including high-performance GPUs and longer runtimes. This investment enabled us to:

1. **Accelerate Training:** Reduced model training times, allowing for more iterations and experiments with different architectures.
2. **Facilitate Collaboration:** Allowed multiple team members to work on the project concurrently, running different experiments and refining the model in parallel.

By leveraging cloud-based resources, we overcame computational bottlenecks and efficiently trained our forecasting models.

Hyperparameters

WeatherNet's forecasting models involved numerous hyperparameters, including:
Learning rate, Batch size, CNN kernel size, Attention Heads, Input Window and more..
Solution: We systematically explored different combinations of hyperparameters by:

1. **Experimentation & Evaluation:** Trained multiple models with varying hyperparameters, evaluating their performance on validation data.
2. **Iterative Fine-Tuning:** Adjusted hyperparameters progressively based on model performance metrics to achieve an optimal balance between accuracy and generalization.

Deployment and User Accessibility

Transitioning WeatherNet from a development environment to a user-friendly web platform required a **robust deployment infrastructure**. We needed to ensure seamless communication between the front-end and back-end while providing an intuitive interface for users to view forecasts.

Solution: We built and deployed a web-based interface that enables real-time interaction with WeatherNet. To ensure a smooth deployment, we utilized **Firebase Hosting** for the front-end and **Firebase Functions** for back-end operations. This serverless setup guarantees scalability, high availability, and minimal maintenance.

We developed the **user interface (UI) with React**, prioritizing an intuitive and user-friendly design. The UI facilitates effortless navigation through real-time weather forecasts, comparison tools, and accuracy insights. To enhance usability, we incorporated **light and dark themes**, allowing users to customize their viewing experience for optimal accessibility.

From Vision to Reality: What We Achieved

Phase A Expectations vs. Achievements

In Phase A, we set ambitious goals for our project, aiming to develop an Israeli weather forecasting website that not only provides real-time temperature predictions but also serves as an informative and research-oriented platform. Our primary objectives included:

- Designing a flexible architecture capable of delivering real-time weather forecasts based on the available data, rather than being restricted to specific locations. Initially, we aimed to focus on Haifa, Tel Aviv, and Beer-Sheva, but due to data limitations in these regions, we applied our model to other stations with better data quality.
- Integrating real-time meteorological data from the Israel Meteorological Service (IMS) to enhance forecast accuracy.
- Developing a website with interactive tools for users to explore our predictions and compare them with traditional IMS forecasts.
- Providing research-oriented insights, including model architecture, prediction accuracy tracking, and forecast comparisons.

What We Achieved

Overall, we successfully met our primary objectives. We developed a robust ML model that predicts temperatures with high accuracy and implemented a functional website that enables users to interact with the predictions, explore the model's architecture, and compare our results with IMS forecasts. Below, we provide a detailed breakdown of our accomplishments.

We will introduce to you two major accomplishments: the development of an advanced machine learning model for weather forecasting and the creation of a website to present our predictions and insights. The goal was to improve the accuracy of temperature forecasting using historical data and provide an accessible, user-friendly platform for viewing and analyzing the results. Below, we detail these achievements.

Predicting Weather Using an ML Model

All the analysis is conducted on the validation set, which the model has not been exposed to during training.

We successfully predicted temperature for the "target station" for the next 2 days, broken down into hourly intervals, using a sophisticated machine learning architecture. This hybrid model combines multiple CNNs, linear layers, transformers, and other techniques. It integrates four specialized sub-models, each optimized for a specific time window, to enhance forecasting accuracy.

Results:

- **0-12 hours:** Maximum error up to **2.65°C**
- **12-24 hours:** Maximum error up to **3.87°C**

In **80% of the cases**, the error remains within **3.05°C**.

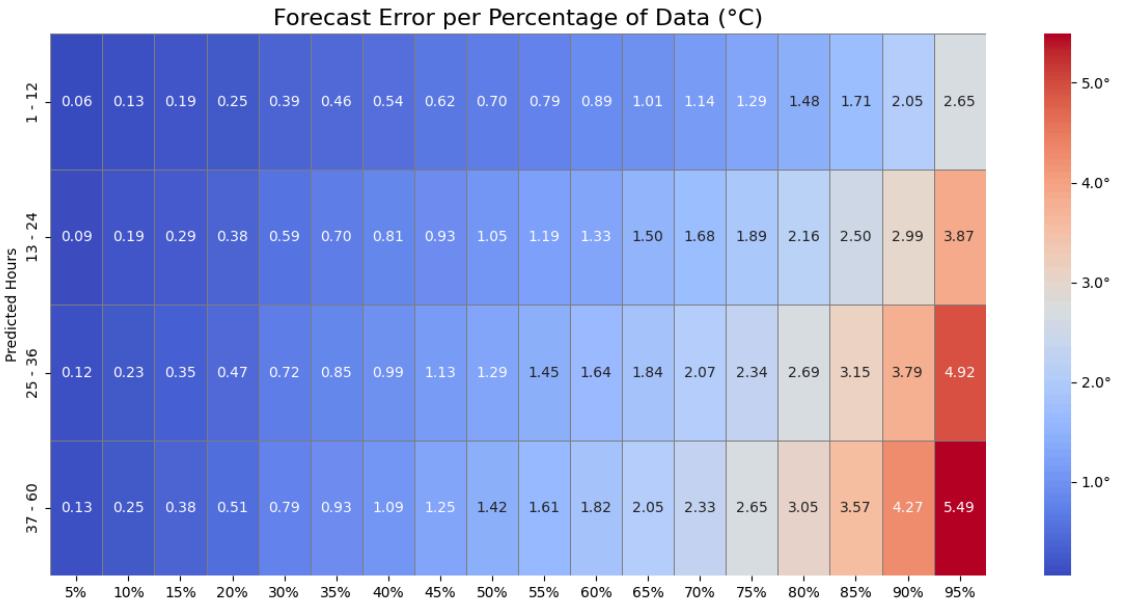

figure 05. Forecast Error Heatmap by Prediction Interval
Note. *Our forecast model predicts temperature at intervals from 1 to 60 hours.*
This heatmap shows the forecast error across these intervals.
Rows represent prediction intervals, and columns indicate the percentage of validation data.
Color intensity reflects error magnitude in degrees Celsius, with darker shades indicating higher errors.

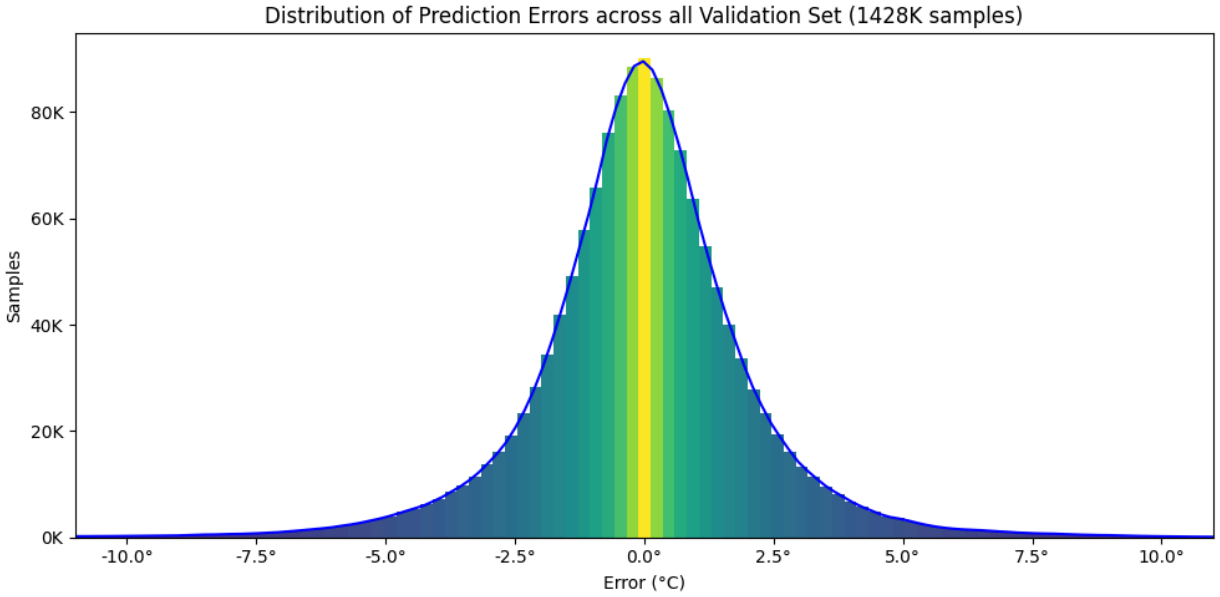

figure 06. Distribution of Prediction Errors
Note. *This graph illustrates the distribution of prediction errors across the entire validation set, comprising vast amount of samples.*
The x-axis represents the error in degrees Celsius, while the y-axis shows the number of samples. The distribution is centered around 0°C, indicating that most predictions are accurate, with errors symmetrically distributed around this point.

Website Development

We wanted to create a website that not only displays our predictions but also allows users to explore our model's architecture and gain insights into its performance. The site serves as an interactive platform for both researchers and general users to understand the workings of our machine learning model and assess its accuracy over different forecasting periods.

We developed a website that provides:

1. Real-time display of our temperature predictions.
2. A historical view of the past few hours.
3. The official IMS (Israel Meteorological Service) forecast for comparison.
4. A combined visualization displaying all three predictions in a single graph for better insights.
5. Additional insights about our model:

- **Architecture page:** An in-depth explanation of our ML model's structure.
- **Statistics page:** Performance evaluation with various graphs, insights, and extensive metrics.

6. Documentation, including:
   - **Phase A submission paper**
   - **User manual**
   - **Developer manual**

Closing Statement

The completion of the WeatherNet project marks a significant achievement in the application of machine learning for weather forecasting. Through rigorous research, iterative development, and continuous refinement, we have successfully delivered a robust and scalable system capable of providing accurate mid-term temperature predictions for Israel. This paper has detailed the journey from initial research and proof of concept to full-scale implementation, showcasing the technical innovations and solutions developed along the way.

We extend our deepest gratitude to the Israel Meteorological Service (IMS) for their invaluable collaboration and support. Their provision of real-time and historical weather data was instrumental in training and validating our machine learning models, ensuring the reliability and accuracy of our forecasts. The partnership with IMS greatly contributed to the overall success of this project.

As WeatherNet reaches its final phase, we take pride in the accomplishments achieved and the knowledge gained throughout this journey. This project not only demonstrates the effectiveness of machine learning in meteorology but also sets a foundation for future advancements in data-driven weather forecasting. We hope that WeatherNet will continue to serve as a valuable tool for researchers, meteorologists, and the broader community, enhancing the accessibility and accuracy of weather predictions.